2. Prime Number Checker

Write a function that checks if a given number is prime.

- Function prototype: `prime(number)`
- Using the `Prime` Class

```
┌──(imen㉿hbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ cat 2-main.rb
require_relative '2-prime'

puts prime(5)
puts prime(6)
puts prime(7)
puts prime(8)
puts prime(9)
puts prime(10)
puts prime(101)
```

```
┌──(imen㉿hbtn-lab)-[…/scripting_cyber/0x00-ruby_scripting]
└─$ ruby 2-main.rb
true
false
true
false
false
false
true
```

Full Code Explanation:

```ruby
# 2-prime.rb

require 'prime'  # Import the Prime class from Ruby's standard library

def prime(number)
  Prime.prime?(number)  # Use the prime? method from the Prime class to check primality
end
```

1. **`require 'prime'`**

- This line imports the `prime` module from Ruby's standard library, which provides functionality related to prime numbers.
- The `Prime` class in this module has methods that allow you to check if a number is prime (`prime?`), generate prime numbers, and perform other prime-related tasks.
- Without this line, you wouldn't be able to use the `Prime` class or its methods in your script.

2. `def prime(number)`

- This line defines a **method** named `prime` that accepts a single argument, `number`. This method will be used to check if the given number is prime or not.
- `def` is the keyword used to define methods in Ruby.

3. `Prime.prime?(number)`

- **`Prime`** is a class that contains methods related to prime numbers. The `prime?` method is used to check if a number is prime.
- `prime?` is a **built-in method** that returns:
  - `true` if the `number` is prime.
  - `false` if the `number` is not prime (i.e., if it's composite or 1).
- This method performs a primality test on the given `number` and returns the result.

4. `end`

- This marks the end of the `prime` method definition. Every method in Ruby must be closed with the `end` keyword.
- **`require 'prime'`**: Makes the `Prime` class available in the script.
- **`def prime(number)`**: Defines a method called `prime` that takes `number` as an argument.
- **`Prime.prime?(number)`**: Checks if the given `number` is prime using the `prime?` method from the `Prime` class.
- **`end`**: Closes the method definition.

Example of How It Works:

- If you call `prime(7)`, the method will check if 7 is prime. Since 7 is prime, the method will return `true`.
- If you call `prime(6)`, it will return `false`, as 6 is not prime (it's divisible by 1, 2, 3, and 6).

Example Usage in `2-main.rb`:

```
require_relative '2-prime'

puts prime(5)    # true, since 5 is prime
puts prime(6)    # false, since 6 is not prime
```

```
puts prime(7)   # true, since 7 is prime
puts prime(8)   # false, since 8 is not prime
puts prime(9)   # false, since 9 is not prime
puts prime(10)  # false, since 10 is not prime
puts prime(101) # true, since 101 is prime
```

When you run this script, it will output:

```
true
false
true
false
false
false
true
```

This script checks if each number is prime and prints `true` or `false` based on the result.